Contents

Istio DNS Certificate Management

3 minute read page test

This task shows how to provision and manage DNS certificates using Chiron, a lightweight component linked with Istiod that signs certificates using the Kubernetes CA APIs without maintaining its own private key. Using this feature has the following

advantages:

- Unlike Istiod, this feature doesn’t require maintaining a private signing key, which enhances security.
- Simplified root certificate distribution to TLS clients. Clients no longer need to wait for Istiod to generate and distribute its CA certificate.

Before you begin

- Install Istio through `istioctl` with DNS certificates configured. The configuration is read when Istiod starts.

```
$ cat <<EOF > ./istio.yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    certificates:
      - secretName: dns.example1-service-account
        dnsNames: [example1.istio-system.svc, example1.istio-sys
tem]
      - secretName: dns.example2-service-account
        dnsNames: [example2.istio-system.svc, example2.istio-sys
tem]
EOF
$ istioctl install -f ./istio.yaml
```

DNS certificate provisioning and management

Istio provisions the DNS names and secret names for the DNS certificates based on configuration you provide. The DNS certificates provisioned are signed by the Kubernetes CA and stored in the secrets following your configuration. Istio also manages the lifecycle of the DNS certificates, including their rotations and regenerations.

Configure DNS certificates

The `IstioOperator` custom resource used to configure Istio in the `istioctl install` command, above, contains an example DNS certificate configuration. Within, the `dnsNames` field specifies the DNS names in a certificate and the `secretName` field specifies the name of the Kubernetes secret used to store the certificate and the key.

Check the provisioning of

DNS certificates

After configuring Istio to generate DNS certificates and storing them in secrets of your choosing, you can verify that the certificates were provisioned and work properly.

To check that Istio generated the `dns.example1-service-account` DNS certificate as configured in the example, and that the certificate contains the configured DNS names, you need to get the secret from Kubernetes, parse it, decode it, and view its text output with the following command:

```
$ kubectl get secret dns.example1-service-account -n istio-syste
m -o jsonpath="{.data['cert-chain\.pem']}" | base64 --decode | o
penssl x509 -in /dev/stdin -text -noout
```

The text output should include:

```
X509v3 Subject Alternative Name:
    DNS:example1.istio-system.svc, DNS:example1.istio-system
```

Regenerating a DNS certificate

Istio can also regenerate DNS certificates that were mistakenly deleted. Next, we show how you can delete a recently configured certificate and verify Istio regenerates it automatically.

1. Delete the secret storing the DNS certificate configured earlier:

```
$ kubectl delete secret dns.example1-service-account -n ist
io-system
```

2. To check that Istio regenerated the deleted DNS certificate, and that the certificate contains the configured DNS names, you need to get the secret from Kubernetes, parse it, decode it, and

view its text output with the following command:

```
$ sleep 10; kubectl get secret dns.example1-service-account
 -n istio-system -o jsonpath="{.data['cert-chain\.pem']}" |
 base64 --decode | openssl x509 -in /dev/stdin -text -noout
```

The output should include:

```
X509v3 Subject Alternative Name:
    DNS:example1.istio-system.svc, DNS:example1.istio-system
```

Cleanup

- To remove the `istio-system` namespace:

```
$ kubectl delete ns istio-system
```